Ann Math Artif Intell (2008) 54:207–222
DOI 10.1007/s10472-009-9137-6

A semiparametric generative model for efficient structured-output supervised learning

Fabrizio Costa · Andrea Passerini · Marco Lippi · Paolo Frasconi

Published online: 5 May 2009

Abstract We present a semiparametric generative model for supervised learning with structured outputs. The main algorithmic idea is to replace the parameters of an underlying generative model (such as a stochastic grammars) with input-dependent predictions obtained by (kernel) logistic regression. This method avoids the computational burden associated with the comparison between target and predicted structure during the training phase, but requires as an additional input a vector of sufficient statistics for each training example. The resulting training algorithm is asymptotically more efficient than structured output SVM as the size of the output structure grows. At the same time, by computing parameters of a joint distribution as a function of the full input structure, typical expressiveness limitations of related conditional models (such as maximum entropy Markov models) can be potentially avoided. Empirical results on artificial and real data (in the domains of natural

F. Costa · A. Passerini · M. Lippi (✉) · P. Frasconi
Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze,
Via di Santa Marta 3, 50139 Firenze, Italy
e-mail: lippi@dsi.unifi.it

F. Costa
e-mail: costa@dsi.unifi.it

A. Passerini
e-mail: passerini@dsi.unifi.it

P. Frasconi
e-mail: paolo@dsi.unifi.it

Present Address:
F. Costa
Departement Computerwetenschappen, Katholieke Universiteit Leuven,
Celestijnenlaan 200 A, 3001 Heverlee, Belgium

Present Address:
A. Passerini
Dipartimento di Ingegneria e Scienza dell'Informazione, Università degli Studi di Trento,
Sommarive st. 14, POVO 38100 Trento, Italy

language parsing and RNA secondary structure prediction) show that the method works well in practice and scales up with the size of the output structures.

Keywords Semiparametric generative model · Supervised learning · Structured outputs

Mathematics Subject Classification (2000) 68T05

1 Introduction

A structured output prediction problem can be formulated as a supervised learning problem where the output or target space is not restricted in any way and can be any set (e.g. a set of graphs or sequences). In this way, several predictions can be made collectively on the same input instance, still maintaining that instances are sampled independently. The supervised learning problem with structured outputs is substantially more difficult to solve compared to the case of scalar outputs (like binary classification or regression) because of the exponential explosion of alternative predictions to be searched. One crucial issue is the choice of the loss function. In many large margin methods, the loss is a *global* measure of the discrepancy between the predicted structure and the desired output [13, 14]. Regularized margin maximization has desirable properties but introduces additional computational costs in the case of structured outputs. In facts, the functional margin in these approaches is a function $f(x) = \arg\max_y F(x, y) = w\psi(x, y)$ of both the input and output portions of the data, x and y, respectively. The arg max computation (necessary during training for verifying margin constraints), is a bottleneck for these approaches and will dominate learning time complexity as the size of the predicted output grows. It is important to stress that combined input and output features may bring relevant information into the learning process, also helping generalization in practice [13]. However, a detailed theoretical analysis shows that the loss functions employed in [13, 14] are not consistent, and that consistent versions of the losses lead to non-convex optimization problems [9].

A somewhat different perspective is explored in kernel dependency estimation (KDE) [3, 15], which focuses on disjoint input and output feature spaces, $\psi(y)$ and $\phi(x)$, respectively. KDE essentially consists of two steps: feature estimation and pre-image calculation. In the first step, the image of the given target $\psi(y)$ is approximated as a function $g(x)$ of the input x. In the second step, the pre-image of $g(x)$ is computed in order to obtain a structured prediction $f(x)$. The second step also involves searching the output space by minimizing a loss between $g(x)$ and $\psi(y)$ with respect to y, but it is only necessary when making predictions and not during learning.

Structured output prediction tasks have been solved for many years using efficient generative models such as stochastic grammars. Two landmark historical applications of stochastic context free grammars (SCFG), also employed in the experimental section of this paper, are natural language parsing (where the input is a sentence and the output a parse tree [2, 8]) and RNA secondary structure prediction (where the input is a sequence of nucleotides and the output a connection graph between nucleotides representing RNA secondary structure [12]). Generative approaches are appealing because of their simplicity and, in practice, their usage is also necessary to

solve the arg max problem in large margin approaches [13, 14]. Of course generative approaches may suffer asymptotic convergence problems when the available background knowledge is incomplete or imperfect and the resulting hypothesized model structure is incorrect. In these situations, conditional models should be preferred to reduce bias error. Computationally efficient conditional models can be obtained by decomposing the prediction task into separate problems associated with output parts. One attempt in this direction is maximum-entropy Markov models (MEMM) [10], which on the other hand are known to suffer from the label-bias problem. Label-bias can be eliminated by evaluating the discrepancy between the whole predicted and target structure, as in the case of large margin methods. This is essentially the approach followed by conditional random fields (CRF) [6].

In this paper, we suggest a simple decomposable model that consist of a set of conditional multinomial distributions associated with an underlying generative model. For example in the case of stochastic grammars, the multinomials would model the conditional probabilities of possible expansions for a given non-terminal symbol, given the input sentence. Input dependency brings a discriminant aspect into the model, which allows us to compensate for imperfect generative structures. Assuming that the sufficient statistics (counts) for the underlying generative model are observed for each example, the structured-output problem can be divided into a set of polytomous scalar sub-problems that can be independently solved by (kernel) logistic regression. In the case of partially observed counts, a (generalized) expectation-maximization approach could be also pursued.

Our experiments suggest that our approach is less affected by label-bias problem of MEMM, while avoiding the expensive global normalization required by CRF. The method can also be seen as an instance of KDE where the output features are the normalized counts of the parts of which the target structure consists and where the pre-image calculation problem is solved by a max propagation algorithm (Viterbi decoding) on the generative model. In the remainder of this paper we show that this method is applicable in several realistic settings, and works well in artificial and real-world problems.

2 Model and algorithms

2.1 General setting

Let $\mathcal{X}$ and $\mathcal{Y}$ be two arbitrary sets representing the input and the output spaces of a supervised learning algorithm. We assume that input-output pairs are sampled from a fixed unknown distribution $p(x, y)$. The input to the learning algorithm consists of: 1) a data set $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots m\}$ of i.i.d. input-output pairs; 2) a generative model with structure S used to build elements of $\mathcal{Y}$ that admits as parameters θ a finite set of M multinomials; 3) for each example (x_i, y_i), a complete vector of sufficient statistics (counts) $c(x_i, y_i)$ for the parameters θ associated with model structure S. The core of the idea is to learn the parameters for a generative machinery as a function of x in order to approximate $p(x, y)$. More formally, the goal of learning is to find a prediction function $f : \mathcal{X} \mapsto \mathcal{Y}$ that approximates the probabilistic dependency between inputs and outputs; we do this by decomposing the problem into two phases and introducing an auxiliary function $g(\cdot)$. Let us denote by Θ the value space for the parameters associated with S and let $g : \mathcal{X} \mapsto \Theta$ be a function that maps each input

object x into a corresponding vector of parameters $\theta = g(x) \in \mathbb{R}^n$. Let $\Pr(x, y; \theta)$ denote the probability of the pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ under the model structure S with parameters θ. We then define

$$f(x) = \arg\max_{y \in \mathcal{Y}} \Pr(x, y; \theta) \tag{1}$$

Learning in this setting only involves searching the function g, while f is computed by an appropriate inference procedure.

2.2 Examples

We now illustrate the above setting with the help of some examples.

Example 1 A typical natural language problem is parsing. In this case, $\mathcal{X}$ is a set of sentences and $\mathcal{Y}$ a set of parse trees. The data set is a collection of sentences each one with its corresponding parse tree. The model structure in this case is a grammar $S = (T, N, P)$, where T and N are the sets of terminal and non-terminal symbols, respectively. In a stochastic *context-free* grammar, rules in P have the form $A_k \mapsto \alpha_\ell$, $A_k \in N$, $\alpha_\ell \in (N \cup T \cup \{\epsilon\})^*$, and have an associated probability $\theta_{k\ell}$. Inference in this case is efficient, but the model cannot capture context dependencies in natural language [8]. Context dependency can be incorporated in the model if production probabilities are computed as function of the input string x, i.e., $\theta_{k\ell} = g_{k\ell}(x)$. This modification does not affect the running time of the inside-outside procedure for computing the most likely parse tree. Sufficient statistics in this case are $c_{k\ell}(x, y)$, counting the number of times a production rule was used to generate example (x, y), and can readily be obtained from the data.

Example 2 SCFGs have been extensively applied to the RNA secondary structure prediction problem [5, 12]. Here we are given a set of strings that encode the base sequences of various types of RNA and we are required to infer their folding structure. This structure can be computed as the parse tree of a SCFG (see Section 4.3), but since each RNA family will exhibit different preferred sub-structures, we can think that data in this domain was actually generated by a mixture of SCFGs, one for each RNA family, sharing the same structure but having different parameters. Learning a SCFG mixture is challenging because the number of underlying components is generally unknown. Rather than introducing a latent variable for modeling family membership, we can use a single structure with input-dependent parameters as in the above setting. This offers the advantage that the number of families does not need to be fixed in advance nor be finite.

Example 3 Modeling sequences with high-order Hidden Markov Models (HMM) allows to capture complex dependencies but is expensive in terms of model parameters. Using the above framework, we can stick to the first-order structure and let parameters depend on the whole sequence, retaining the ability of modeling higher-order dependencies. This can be seen as a way to use the similarity between sequences to approximate the memory properties of a more complex generative model. Note, however, that a first-order HMM with data-dependent parameters cannot approximate a second order HMM with arbitrary accuracy; consider in fact the ambiguity that is introduced when two different two-state transitions are collapsed

into the same one, but both occur in some of the observation sequences. In these cases the first-order HMM has to collapse two potentially different probability distributions into a single one, and the two cases can not be disambiguated.

2.3 Learning and inference

We assume that the model structure S is characterized by a disjoint set of multinomial distributions and denote by θ the corresponding set of parameters.[1] For each multinomial distribution k, having size n_k, let $\theta_k = g_k(x) = \sigma(W_k' \phi(x))$, where $\phi(x) \in \mathbb{R}^d$ is a feature vector for x, $W_k \in \mathbb{R}^{d \times n_k}$ a weight matrix, and σ the normalized exponential function:

$$\sigma_\ell\left(W_k' \phi(x)\right) = \frac{\exp\left(\left(W_k'\right)_\ell \phi(x)\right)}{\sum_{j=1}^{n_k} \exp\left(\left(W_k'\right)_j \phi(x)\right)} \tag{2}$$

where $(M)_i$ stands for the i^{th} row of M. Since sufficient statistics for θ are fully observable in the training data, learning does not require inference and reduces to finding the weight matrices W_k in a discriminative setting. Each multinomial k can be independently learned by maximizing its regularized log-likelihood to the normalized counts in the training set:

$$\arg\min_{W_k} \sum_{i=1}^{m} \sum_{\ell=1}^{n_k} \theta_{k\ell}(x_i, y_i) \log \sigma_\ell\left(W_k' \phi(x_i)\right) + \mu C[W_k] \tag{3}$$

where μ is a regularization constant and $C[W]$ is a penalty measuring the complexity of W. A standard choice, which was made in all reported experiments, is that of penalizing non uniform distributions, obtained by setting $C[W] = \|W\|^2$.

Assuming a valid input kernel function $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is available, g_k can also be written

$$g_k(x) = \sigma\left(\sum_{i=1}^{m} \alpha_i \kappa(x, x_i)\right) \tag{4}$$

where $\alpha_i \in \mathbb{R}^{n_k}$ are the solution coefficients determined by the kernel logistic regression [16] problem arising from the dual of (3).

The procedure for inferring the output structure of an example x is the following: first, the learned (kernel) logistic regression models are used to compute example-dependent parameters of the multinomials; then, the most-likely output structure is inferred maximizing the probability in (1). Polynomial time algorithms for exact inference exist for a number of cases, such as Viterbi parsing for SCFGs. Approximate inference techniques can be employed when exact inference is intractable.

2.4 Complexity

It is crucial to observe that the prediction function $f(x)$ is never computed during the learning phase, thanks to our assumption that sufficient statistics $c(x, y)$ are observable.

[1] In the NLP example, the multinomials are the probability distributions of the possible expansions of each non-terminal symbol allowed by the grammar.

We have to solve a number of logistic regression problems equal to the number $\mathcal{M}$ of multinomials in the model. It can be shown that each of these problems has a complexity which grows with the square of the number m of training examples [16] and linearly with the number of values of the multinomial. The overall complexity is thus $O(m^2K)$ where K is the sum of the number of values of all multinomials; in the case of SCFGs, for example, this is equal to the number of production rules $|P|$, hence the complexity results in $O(m^2|P|)$.

3 Related works

The proposed approach can be seen as an instance of KDE [15]. By introducing feature mappings $\phi(x)$ and $\psi(y)$ for inputs and outputs respectively, KDE first learns the mapping $x \rightarrow \psi(y)$ by kernelized vector-output regression, and then solves the pre-image problem of computing the y whose image is the most similar to the predicted image $\psi(y)$ under a certain norm. In [3], the pre-image problem was addressed by a graph theoretical algorithm for the case of output strings and k-gram output kernels. In our approach the output image $\psi(y)$ is the vector of normalized counts of output structure parts, and the pre-image problem is solved using an inference algorithm on the generative model (i.e., the Viterbi algorithm in the case of Natural Language Parsing).

From the perspective of probabilistic graphical models, our algorithm extends reparametrization of generative models to arbitrary sets of multinomials, while exploiting complex input dependency via the kernel function. By re-estimating parameters of the joint model as a function of the full input structure, the approach should be less affected by the label bias problem [6] of MEMM [10], which is due to the local (multinomial-wise) normalization, while avoiding the expensive global normalization required by CRF. Artificial experiments in Section 4.1 seem to corroborate such intuition, as our algorithm consistently outperforms those generative models (HMM and SCFG) which are used to approximate more complex underlying ones. The first setting—approximating a second order HMM with a first order model—is actually very similar to the one shown in [6], where MEMM were outperformed by first order HMM.

Collins [2] introduced the use of margin-based approaches to parameter estimation in structured output prediction. Other researchers [13, 14] have proposed different loss functions weighting the structural difference between predicted and true output. All approaches based on large margin have to face the problem of an exponential number of constraints (variables in the dual formulation). Taskar et al. [13] addressed the problem by marginalizing them over the cliques of the Markov network. Tsochantaridis et al. [14] employed an iterative procedure, adding the single most violated constraint and retraining the model at each iteration, until no constraint was violated by more than a given threshold. The most violated constraint is found solving a maximization problem ($\arg\max$) for each training example; if the violation is over a certain threshold, a new SVM is then trained starting from the previous solution and adding the new constraint. The authors prove that the global number of constraints added—and thus of SVM optimization problems to be solved—is linear with the number m of training examples. The maximization problem corresponds to the inference task described in Section 2.3. In the case of

SCFGs, for example, it can be computed via Viterbi parsing, so its running time is cubic in the length L of a given sentence and dominates the running time of the algorithm, as pointed out by the authors [14]. It must be remarked that our method, on the contrary, does not need to perform inference during learning, and this may become a crucial factor when the data set contains large input structures (for example, in the case of RNA sequences).

McAllester [9] showed that the loss functions employed in [2, 13, 14] are not consistent, and that consistent versions of the losses lead to non-convex optimization problems. This trade-off between convexity and consistency suggests that additional restrictive assumptions may be usefully explored when learning with structured-output data. Unlike the above methods, our solution does not employ a "global" loss between target and predicted structures, but a measure of discrepancy between the observed and the predicted multinomial distributions associated with "local" building steps in the generative process. Only in very special cases it is easy to prove consistency, for example under the assumption that at most one of the production rules with the same left-most part is used in each output structure; otherwise, it remains an open problem to determine practically significant and more general hypotheses under which our approach guarantees consistency.

4 Experiments

4.1 Artificial experiments

We run a number of artificial experiments to show how data generated by large context free grammars can be successfully learned by a simpler SCFG whose parameters depend on the input. Specifically, for simpler SCFG we mean here a model that has a reduced alphabet set—and consequently a reduced rule set—than the one which generated the data. All the artificial data sets are generated using the Simple Language Generator[2] (SLG). In the remainder of the paper, we will refer to our method as Kernel Logistic Regression for Structured Output (KLRSO).

4.1.1 Approximating second-order hidden Markov models

The first artificial experiment consists in label sequence learning and is similar to the one used to show the label-bias problem in MEMM [6]: the true generative model is a second-order HMM, where transition rules depend on the previous state in addition to the dependency on the current state, while the simplified model is a standard HMM.

Note that it can be readily shown how a second order HMM can be represented with a regular grammar: a second order HMM with n states can be equivalently expressed with a regular grammar with n^2 non-terminal symbols and with a rule for each pair state-transition and emission-transition. Using standard HMMs means that we use a regular grammar with n states instead of n^2 and a correspondingly smaller number of rules. This implies that a simple HMM collapses several states

[2] http://www.cs.cmu.edu/~dr/Projects/SLG

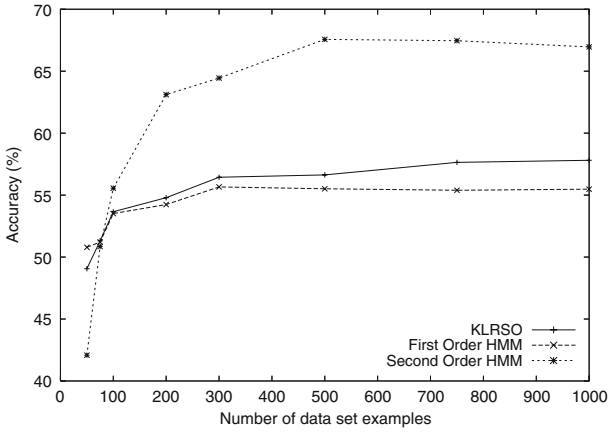

Fig. 1 Approximation of a second order HMM by a first order HMM and by the KLRSO model as the number of data points increases

and rules and does not have enough expressive power to represent all transitions/emissions of the true generative model. In the approach presented, however, we learn the parameters of the simpler regular grammar using the whole string, so that similarity between sequences can approximate the memory properties of the real generative model.

We run a set of experiments comparing KLRSO with both first and second order HMM for data sets of increasing size. Input features for KLRSO are provided by a spectrum kernel [7] working on k-mers of variable size (3–7) composed with a third degree polynomial kernel. Figure 1 reports learning curves for the three methods, where results at each point are averaged over ten data sets randomly generated by second-order HMM. As expected, we observe an increase in accuracy for more complex models (the second order HMM and the KLRSO) as more data points become available for parameters estimation, and a higher accuracy for the KLRSO in respect to the simple HMM. The difference between first-order HMM and KLRSO is statistically significant with p-value equals to 0.016, when using a data set of 1000 examples.

4.1.2 Approximating context-free grammars

The second artificial experiment consists in approximating a complex stochastic context-free grammar with a simpler one. We studied two cases: in the first one, the

Table 1 First task for artificial grammars: the true generative model is the union of n grammars sharing the same structure

n	SCFG	KLRSO	SCFG	KLRSO
2	64.04 ± 1.67	70.09 ± 1.00	57.54 ± 2.39	67.33 ± 1.81
3	63.24 ± 2.13	67.70 ± 1.42	62.79 ± 3.69	65.53 ± 3.10
4	64.04 ± 2.05	68.72 ± 2.09	62.28 ± 2.66	65.59 ± 3.04
	F-Measure		Complete Match	

Results obtained by our method are compared to a simple context-free grammar learned on the training set. Different rows correspond to a different number of grammars n; results on each row are averaged over five different experiments. Differences are statistically significant with p-value < 0.01

Table 2 Second task for artificial grammars: the real generative model is composed by two SCFGs having different structures

m	SCFG	KLRSO	SCFG	KLRSO
100	43.67 ± 3.63	44.94 ± 2.61	50.00 ± 4.80	52.94 ± 4.80
300	46.23 ± 1.30	51.88 ± 1.55	49.67 ± 2.36	62.33 ± 2.49
500	48.75 ± 0.15	52.54 ± 0.90	51.79 ± 0.43	62.89 ± 1.28
	F-Measure		Complete Match	

Results obtained by our method are compared to a simple context-free grammar learned on the training set. Different rows correspond to different training set sizes m; results on each row are averaged over three different experiments. When $m = 100$ the differences are not statistically significative, while when $m = 500$ they are, with p-value < 0.01

true generative process is the union of n SCFGs which share the same structure (i.e., the same rule set) but have different parameters (i.e., different rule probabilities). The union grammar chooses, with a specified probability distribution, one of the grammars to generate a sentence. In this setting we approximate the union with a single base grammar but, once again, we allow the parameters to depend on the generated string. We see this task as an artificial model reflecting what happens in the RNA Secondary Structure Prediction Task (see Section 4.3). In the second experiment, the true generative process is the union of two grammars that do not share the same structure. Here our simpler model S is the union of all rules of each distinct grammar.

For both experiments, we compared a standard SCFG with parameters estimated from the training data set, to our method with a polynomial kernel of degree three on top of a spectrum kernel. Results shown in Tables 1 and 2 indicate that the proposed approach models the underlying generative process more accurately than a standard SCFG. For the second experiment, results are reported for different values of the training set dimension m: it is interesting to observe that when m is too small, the difference between our method and a SCFG is not statistically significant; on the contrary, when m increases, the difference is statistically significant with p-value < 0.01, which is due to a better approximation of the probability distributions of the sufficient statistics.

4.2 Natural language parsing

Parsing is a fundamental task in Natural Language Processing (NLP) and is one of the most studied structured output learning problems. In parsing we are given a variable length sequence of words (a sentence) $x = x_1, \ldots, x_k$, and the goal is to build the (most plausible) explanation y of x. The explanation usually takes the form of a derivation tree, that is, a tree structure that represents the order in which certain rules from a formal grammar are applied to derive the sentence [2, 4]. Although human language cannot be fully explained by simple SCFGs, these are usually adopted as a

Table 3 Results on 163 short sentences of Penn TreeBank (length ≤ 10)

	SCFG	$svm^{cfg}(0/1)$	$svm^{cfg}(F_1)$	KLRSO
F-Measure	84.07	86.36	87.34	87.32
Complete Match	55.97	60.25	59.63	63.35

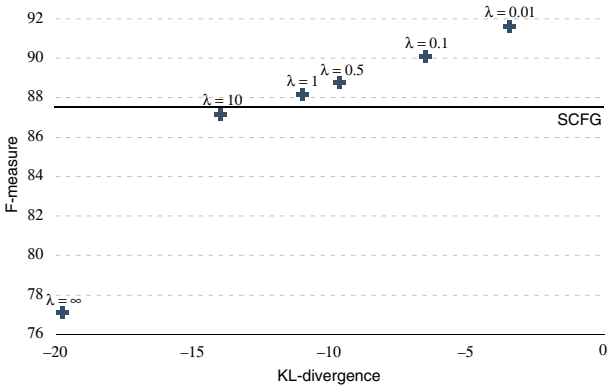

Fig. 2 F-Measure performance on training data, function of the Kullback-Leibler divergence between the predicted distribution and the one observed on training data. After performing the training, we compute the sum, over all the heads of the grammar, and over all the training examples, of the Kullback-Leibler divergence between the observed distribution for that example, and the predicted one. Each point is calculated using as the target distribution a different distribution $\hat{P} = \lambda P + (1 - \lambda)U$, where P is the distribution of the observed counts, and U is the uniform distribution; the values of λ which have been used are mentioned in the figure

good approximation. Here we propose to decompose the problem of learning the probabilities of SCFG rules in M different problems, where M is the number of heads[3] of the grammar, on a per example basis.

We performed three experiments: in the first one (proposed by [14]) we predict the model probabilities over short sentences in the Penn TreeBank, in the second one we predict only rules involving verbs and in the last one we worked on Italian sentences from the Turin University TreeBank. Finally, we report some experiments concerning the running time of our approach with respect to the length of the input, in comparison with svm^{cfg} by Joachims.

4.2.1 Parsing short sentences of Penn TreeBank

Our first experiment on NLP reproduces the one proposed by [14]: we select 4,071 sentences of length at most 10 from section F2–21 of the Penn TreeBank Wall Street Journal corpus as training set, and 163 sentences of length at most 10 from section F22 as test set. The grammar is processed[4] as in [4] yielding 1,517 rules and 27 logistic regression problems (one per head). We further limit the rules involved in the prediction task as some heads (NP, VP, S) induce logistic regression problems with high cardinality (e.g. 547 for NP). This saves computational time and also increases performance (possibly because of a beneficial effect in reducing the sparseness over the regression classes). We therefore use a frequency threshold to limit the number of rules and we resort to the SCFG probabilities estimated on the global training set for the infrequent rules. A further improvement may include the use of

[3]We name *head* the left-most part of a production rule.

[4]Since the process is not explained in full details, we cannot assure that the grammar obtained is exactly the same as the one used by [14].

Table 4 Results on 527 sentences of Penn TreeBank selected for the experiment on the distribution of VP expansions

	SCFG	KLRSO	KLRSO$_{best}$
F-Measure	70.52	70.99	71.43
Complete Match	17.27	21.44	23.72

The third column shows the results which would be achieved by our method, if an oracle predicted the real distribution of VP expansions

a gating network to have a mixture of the two grammars (the context-free and the predicted one).

We tried different kernels over sentences and finally resorted to the Weighted Decomposition Kernel [11] on the sequence of Part-of-speech tags (i.e., discarding the lexical information), with a selector of size 1 and a context of size 5. We compared our method with a SCFG and with the max-margin structured output method svm^{cfg} by Joachims, trained using both F_1 and 0/1 losses. Results are shown in Table 3. Note that the reported results differ from those in [14] as we now do not include the prediction of the root node (which is trivially present in all parses) in the F-Measure.[5]

Figure 2 shows the relation between F-Measure and Kullback-Leibler divergence between the predicted distribution and the one observed on training data. This experiment justifies the choice of the loss function optimized by the logistic regression algorithm, since it shows that minimizing the discrepancy between the sufficient statistics observed on training data and the predicted ones, results in minimizing also the error on the structured output prediction.

4.2.2 Disambiguating VP rules

In the second experiment we predict only the probability distribution of the expansions of the VP head, resorting to the SCFG probabilities for all the other rules. This experiment shows that improvement can be achieved even when dealing with specific ambiguous grammatical phenomena rather than tackling the whole parsing task. We extracted from the whole Penn TreeBank the trees containing only those VP rules appearing at least 10 times in the training sections: we obtained 9,564 training examples from sections F2–F21[6], and a test set of 527 instances from section F22. The number of total VP expansions in this subset was reduced to 46. In Table 4 we report comparative results between our method, a SCFG learned on the whole training set, and a SCFG having the exact VP distribution (which is an upper bound on the performance of our method): the latter column shows that our method recovers most of the gap between the performance of the SCFG and that of the oracle informed about the true VP distribution.

4.2.3 Parsing Italian sentences

The third experiment uses the Turin University Italian TreeBank (TuT) [1]. Despite the small size of this data set (1,790 sentences), we report comparative results to show

[5]This is a common practice and is default in the standard evaluation software *evalb* (http://nlp.cd.nyu.edu/evalb/).

[6]This time we have no limitation on the length of the sentences.

Table 5 Results obtained performing a 5-fold cross validation on the 1,790 examples of the Turin University TreeBank

	SCFG	KLRSO
F-Measure	60.71 ± 1.60	62.50 ± 1.49
Complete Match	12.68 ± 1.27	15.35 ± 1.82

Columns show the F-Measure bracketing and the Complete Match, averaged on the five folds. Differences are statistically significant with p-value < 0.01

how our approach does not require any hand tuning when dealing with different languages. In Table 5 we report the average results on a 5-fold cross validation experiment, comparing our algorithm to a SCFG. For the Italian language, the kernel which gave best results was again a Weighted Decompositional Kernel, but with a selector of size 2 and a context window of size 5.

4.2.4 Complexity issues

In studying complexity issues, it must be remarked that in the method by [14], the cost for computing the argmax becomes dominant when the dimension of the input is large. Furthermore, its effectiveness and efficiency are strongly affected by the choice of the regularization parameter. For the experiments described in the previous section, the best value of C for svm^{cfg} was 10^3; anyway, for measuring temporal performance, we set the value of C to 10^2.

We collected four different data sets, all composed by 100 sentences extracted from the Penn Treebank, each containing sentences whose lengths belong to a specific range: the first set contains only sentences having at most 10 words, the second only sentences containing a number of words between 10 and 20, the third between 20 and 30, and the fourth between 30 and 40. As shown in Table 6, the dependence on the length of the sentences is evident for svm^{cfg}, and this can become a bottleneck when the algorithm has to deal with very long input sequences, as in the case of RNA secondary structure prediction (see Section 4.3).

As for the dependence on the data set dimension, our experiments showed an advantage for svm^{cfg}, when dealing with data sets of short sentences (i.e., 2 min against 6 min with a data set of 1000 examples). Yet, when large data sets contain sentences of any length, the inference performed during the training phase becomes dominant for svm^{cfg}: a final experiment on a set of 5,000 randomly selected sentences showed a marked difference between 7 h for our approach and 2 days and a half for the max-margin method.

Table 6 Time comparison of the two algorithms with inputs of different length

Length	svm^{cfg}	KLRSO
10	3 s	9 s
20	2 min 3 s	17 s
30	1 h 7 min	23 s
40	60 h 44 min 7 s	38 s

Each training set is composed by 100 sentences of length within [L-9,L]. C for svm^{cfg} is set to 100

Table 7 Results averaged on 5-fold cross validation on two RNA families

Family	SCFG	$SCFG_2$	KLRSO	$KLRSO_2$
Enterovirus5CRE	94.30 ± 1.09	90.18 ± 1.23	96.01 ± 1.29	95.85 ± 1.34
Flavivirus	95.55 ± 1.20	84.34 ± 1.95	98.01 ± 0.87	98.16 ± 0.99
Weighted	94.84 ± 1.14	87.67 ± 1.54	96.87 ± 1.11	96.84 ± 1.19

4.3 RNA secondary structure prediction

RNA secondary structure is composed of double-stranded RNA regions formed by folding the single-stranded molecule back on itself: such foldings are made possible by the presence of complementary subsequences in the downstream and upstream of the sequence, so that Watson-Crick base pairing between the complementary nucleotides G/C and A/U can occur.

The prediction of RNA secondary structure consists in finding base-paired regions (stems) and non paired ones (loops) within a given RNA sequence. The use of SCFGs for this kind of task has been studied in several works [5, 12]. Unlike the Natural Language Parsing case, for RNA we do not have the true parse nor the structure of the grammar: the only available information is the base-pairing (called bracketing). In a sequence with a loop *uuggaaa* and a stem, for example, we are given the following pairing information: $\ldots g[u[a[g[c[auuggaaau]g]c]u]a]ugua \ldots$.

As our approach assumes to have direct knowledge of the counts of the rules applied in the derivation of each training string, we need to create a grammar and pre-process the available bracketed strings to infer their most plausible derivation under the chosen grammar. Following the approach described in [5], we assume the grammar structure to consist of the following non-terminal symbols: S for *stem*, L for *loop*, I for *undifferentiated*, and A, C, G, U for the four bases. Rules in this grammar code for the transition from S to L through I and vice-versa allowing the recursive nesting of loops and stems, and constrain stems to produce paired subsequences with rules such as $S \rightarrow ASU$.

To enhance the expressive power of the grammar (and therefore also parsing performance), we extended the grammar rules by splitting each non-terminal T in a certain number k of new non-terminals, where k is a parameter of the new grammar: symbol S for stem, for example, splits up in $S_1, \ldots, S_k$, and the same happens for other non-terminals. Initial symbol becomes I_0, and each undifferentiated symbol I_j can produce a different non-terminal of superior order, that is for example $I_j \rightarrow S_{j+1}$, while S_j and L_j produce non-terminals of the same level: in this way, subscripts of the non-terminals can be seen as a sort of "nesting factor" within the parse tree. This method increases the number of rules of the grammar, but at the same time it enhances its expressivity, since a certain rule, say $S \rightarrow ASU$, can have

Table 8 Results averaged on 5-fold cross validation on three RNA families

Family	SCFG	$SCFG_3$	KLRSO	$KLRSO_3$
Enterovirus5CRE	94.30 ± 1.09	90.08 ± 1.02	96.01 ± 1.29	96.11 ± 1.39
Flavivirus	95.55 ± 1.20	56.56 ± 3.14	98.01 ± 0.87	98.09 ± 1.44
U5SpliceosomalRNA	85.11 ± 1.26	80.32 ± 1.78	86.34 ± 1.48	86.71 ± 1.39
Weighted	91.83 ± 1.18	77.00 ± 1.89	93.61 ± 1.22	93.79 ± 1.41

Table 9 Results averaged on 5-fold cross validation on four RNA families

Family	SCFG	$SCFG_4$	KLRSO	$KLRSO_4$
Enterovirus5CRE	94.30 ± 1.09	88.95 ± 1.58	96.01 ± 1.29	95.85 ± 1.67
Flavivirus	95.55 ± 1.20	53.80 ± 2.74	98.01 ± 0.87	98.16 ± 1.30
U5SpliceosomalRNA	85.11 ± 1.26	77.41 ± 2.67	86.34 ± 1.48	86.26 ± 1.03
HepatitisCVirusVII	70.63 ± 4.51	48.78 ± 3.37	74.41 ± 3.28	74.40 ± 4.51
Weighted	88.05 ± 1.77	70.05 ± 2.46	90.20 ± 1.59	90.17 ± 1.92

different probabilities according to different depths in the tree: $p(S_j \rightarrow AS_jU) \neq p(S_k \rightarrow AS_kU)$, if $k \neq j$.

In order to obtain the true parse trees consistent with the available bracketing information, we incorporate the bracketing symbols in the grammar. The brackets are then removed from the grammar and the sequences together to their output structural information are now coded in the parse tree.

For our experiments we used data collected from the Rfam data set[7], which contains RNA sequences from different families, both manually and automatically annotated. We collected data (both manually and automatically annotated) from four families: Enterovirus5CRE, Flavivirus, HepatitisCVirusVII and U5SpliceosomalRNA. Note that we eliminated sentences with gaps and symbols other than [*acgu*]. The data set obtained was composed by 196 sequences for Enterovirus5CRE, 150 for Flavivirus, 154 for U5SpliceosomalRNA and 108 for HepatitisCVirusVII, for a total of 608 examples.

We run two sets of experiments: in the first one, each family was modeled independently and we compared our method to a standard SCFGs; in the second one, we jointly modeled the families all together, and we learned a single grammar for the whole training set. The latter is particularly interesting, since we do not need to know how many families compose the data set, and which family each sequence belongs to, but we only use the kernel function to measure similarity between different sequences. The kernel employed was a spectrum kernel with k-mers of size 5 up to 9, composed with a third degree polynomial kernel. The experiments were run first on the first two families (Enterovirus5CRE and Flavivirus), then we added the third family (U5SpliceosomalRNA) and finally the fourth (HepatitisCVirusVII). For all the experiments we performed a 5-fold cross validation.

All experiments confirm that our method always outperforms the single SCFG learned on each single family; in addition, our second set of experiments shows that even a grammar learned on the whole data set, composed by all the families, outperforms the SCFG learned on the single families. As a comparison, we even learned a single SCFG on all the families: as expected, it performs very poorly, since the families are very different. Results are shown in Tables 7, 8 and 9: columns labeled SCFG and KLRSO report results for models trained on each family separately, while models in $SCFG_j$ and $KLRSO_j$ are trained on j families altogether.

Unfortunately we cannot compare our method against svm^{cfg}, since the current software implementation doesn't allow to easily insert complex input dependent features: in the case of RNA, $\psi(x, y)$ features alone are not expressive enough, preventing svm^{cfg} from learning a relevant grammar for this task.

[7] http://www.sanger.ac.uk/Software/Rfam/

In some cases we noticed that parsing strings of a specific family with the grammar learned from all the families together produces better results than those achieved by a grammar learned on the single family: although the property is not exhibited by all families, it may indicate a sort of information transfer between some families and will be subject of further investigation.

5 Conclusions

We have presented an efficient semi-parametric approach to the structured-output learning problem. The core idea is to learn the parameters of a grammar structure on a per-example basis and use them to guide the inference phase producing the desired output structure. The presented approach exhibits a lower computational cost with respect to maximum-margin based methods by avoiding any inference step during the learning phase. Despite its simplicity the method shows comparable experimental results to max-margin methods while reducing the computational time when the input size becomes large, as it happens when dealing with RNA sequences or with unrestricted natural language sentences.

The proposed approach assumes knowledge of the sufficient statistics of the generative model for each training example. A more general learning setting is conceivable in case of partially observed counts, by relying on a generalized expectation-maximization approach.

References

1. Bosco. C., Lombardo, V., Vassallo, D., Lesmo, L.: Building a treebank for Italian: a data-driven annotation schema. In: Proceedings of the Second International Conference on Language Resources and Evaluation LREC, pp. 99–106, Athens, 31 May–2 June 2000
2. Collins, M.: Parameter estimation for statistical parsing models: theory and practice of distribution-free methods. In: New Developments in Parsing Technology, pp. 19–55. Kluwer Academic, Norwell (previusly IWPT 2001) (2004)
3. Cortes, C., Mohri, M., Weston, J.: A general regression technique for learning transductions. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 153–160, Bonn, 7–11 August 2005
4. Johnson, M.: PCFG models of linguistic tree representations. Comput. Linguist. **24**(4), 613–632 (1998)
5. Knudsen, B., Hein, J.: RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. Bioinformatics **15**(6), 446–454 (1999)
6. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco (2001)
7. Leslie, C., Eskin, E., Noble, W.: The spectrum kernel: a string kernel for svm protein classification. In: Proc. of the Pacific Symposium on Biocomputing, pp. 564–575, Lihue, 3–7 January 2002
8. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT, Cambridge (1999)
9. McAllester, D.: Generalization bounds and consistency for structured labeling. In: Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., Vishwanathan, S.V.N. (eds.) Predicting Structured Data. MIT, Cambridge (2007)
10. McCallum, A., Freitag, D., Pereira, F.C.N.: Maximum entropy markov models for information extraction and segmentation. In: Langley, P. (ed.) ICML, pp. 591–598. Morgan Kaufmann, San Francisco (2000)

11. Menchetti, S., Costa, F., Frasconi, P.: Weighted decomposition kernels. In: Proceedings of the Twenty-second International Conference on Machine Learning (ICML'05), pp. 585–592. ACM, New York (2005)
12. Sakakibara, Y., Brown, M., Hughey, R., Mian, I.S., Sjölander, K., Underwood, R.C., Haussler, D.: Stochastic context-free grammars for tRNA modeling. Nucleic Acids Res. **22**, 5112–5120 (1994)
13. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: Advances in Neural Information Processing Systems (NIPS 2003), Vancouver, 13–18 December 2004
14. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. **6**, 1453–1484 (2005)
15. Weston, J., Chapelle, O., Elisseeff, A., Scholkopf, B., Vapnik, V.: Kernel dependency estimation. Adv. Neural Inf. Process. Syst. **15**, 873–880 (2003)
16. Zhu, J., Hastie, T.: Kernel logistic regression and the import vector machine. In: Advances in Neural Information Processing Systems (NIPS 2001), pp. 1081–1088, Vancouver, 3–8 December 2001